



FlexTree: visualising large quantities of hierarchical information

Song, H., Curran, E., & Sterritt, R. (2002). FlexTree: visualising large quantities of hierarchical information. In *Unknown Host Publication* (Vol. 7). IEEE. <https://doi.org/10.1109/ICSMC.2002.1175707>

[Link to publication record in Ulster University Research Portal](#)

Published in:
Unknown Host Publication

Publication Status:
Published (in print/issue): 01/10/2002

DOI:
[10.1109/ICSMC.2002.1175707](https://doi.org/10.1109/ICSMC.2002.1175707)

Document Version
Publisher's PDF, also known as Version of record

General rights
Copyright for the publications made accessible via Ulster University's Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The Research Portal is Ulster University's institutional repository that provides access to Ulster's research outputs. Every effort has been made to ensure that content in the Research Portal does not infringe any person's rights, or applicable UK laws. If you discover content in the Research Portal that you believe breaches copyright or violates any law, please contact pure-support@ulster.ac.uk.

FlexTree: Visualising Large Quantities of Hierarchical Information

Hongzhi Song Edwin P. Curran Roy Sterritt

School of Information Systems, University of Ulster
Jordanstown, Newtownabbey, Northern Ireland, BT37 0QB, UK
{gh.song, e.p.curran, r.sterritt}@ulster.ac.uk

Abstract—Visualising large quantities of hierarchical information is a difficult topic in Information Visualisation and it has been attracting much effort since the emergence of this research area. This paper presents the FlexTree, a novel approach to visualising, navigating and analysing large hierarchies. It is based on the focus+context technique and combines the power of histogram with traditional two dimensional (2D) node-link diagrams. This approach maintains the context of a large hierarchy while providing easy and consistent access to details of multiple focal points. Simple aesthetic rules and an interactive design were applied to the system. As a demonstration of the approach a computer file system hierarchy with 6,351 file folders and 130,400 files on a personal computer has been successfully visualised.

Keywords—Human-Machine Systems, Information Visualisation, Hierarchy Visualisation, Focus+Context

I. INTRODUCTION

One of the active fields in Information Visualisation research is hierarchy visualisation. A hierarchy, mathematically abstracted as a tree, is a natural means of organising and managing knowledge. Hierarchical structures are ubiquitous: such as computer file system structures, organisational structures, family trees, catalogues, computer programs etc. Thus it is widely used in human-machine systems. If a hierarchy can be extracted from a certain knowledge domain, the structure, patterns, and relationships may be easily seen, and thus more insight into this domain can be acquired. The number of nodes in a balanced tree increases exponentially with the depth of the tree. This feature makes it possible to manage the increased information consistently in the same manner and with a theoretically unlimited number of nodes contained. However, the direct disadvantage of this feature is that it makes it difficult to cope with using traditional visualisation approaches (e.g. node-link diagrams, indented lists) when the amount of information becomes large.

FlexTree, an interactive visualisation approach to visualising hierarchical information is proposed. The following aspects are the aims of the design:

- **Efficient space utilisation:** in order to visualise a large amount of information in a limited screen size; efficient use of space is critical.
- **Interactivity:** easy to use and intuitive controls are necessary to achieve a good user acceptance.
- **Simplicity:** a simple yet powerful system is always a major aim of designers.

- **Completeness:** to keep a complete structure while providing smooth access to details is essential to focus+context based visualisations.
- **Organisation:** well-organised information is always easier to understand.
- **Dual Function:** visual representation is easy to understand whereas textual information is easy to deliver. Providing both of these two forms of information in parallel would improve usability.

II. PROBLEMS AND CHALLENGES

Hierarchies contain two kinds of information: structural (organisational) information associated with the hierarchy and content information associated with individual nodes [7]. To represent both of these types of information for a large hierarchy is a non-trivial task. Johnson and Shneiderman [7] highlight the weakness of traditional methods. They classified traditional methods for presenting hierarchically structured information into three categories: listings, outlines (sometimes referred to as indented lists) and treed diagrams (also called node-link diagrams). Due to user familiarity we focus on the indented list to discuss some major problems, as well as then node-link diagrams, then a natural ancestor to the FlexTree approach.

A. Indented lists

One of the traditional methods used to represent hierarchies is the popular technique of indented lists. A typical example is the Windows Explorer shown in Figure 1 (other file management tools have similar functionality). It presents a computer file system hierarchy using the overview+detail technique with file folders or directories listed on the left panel as the overview and files contained inside a certain folder listed on the right panel as detail. A file can be accessed after locating its folder. However, to find a file located in a further level of the hierarchy, the user needs to traverse all the intermediate folders unless the search facilities are being used. Certainly, search is an alternative way to perform the same task. Nonetheless, searching is a direct-to-point strategy with very little or no contextual information. Nowadays, a personal computer normally has a 10-30 GB disk, so that the file system hierarchy often has 10-20 levels. Furthermore, users tend to store their files into sub-level folders since rearrangement usually happens in the bottom level folders. This makes the hierarchy more complex, and thus imposes on the daily file operations a large amount of overhead. Albeit shortcuts are provided to reduce

such overhead for frequently accessed files, it may be considered as compensation rather than a radical solution.

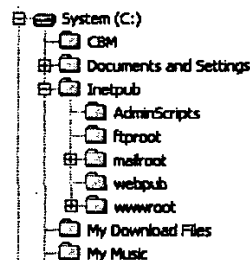


Figure 1. An indented list (Snapshot of Windows Explorer)

Another problem is, when the hierarchy is large, it is not easy for users to keep the structural information in mind. For instance, how could one point out which folders are in the same level and how many folders are in that level? How many levels are contained in this hierarchy?

B. Node-link diagrams

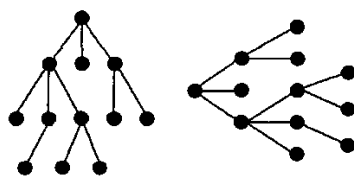


Figure 2. Traditional node-link diagrams

Traditional node-link diagrams (Figure 2) are generally taken as excellent visualisation tools for small trees [12]. However these diagrams make poor use of display space because large percentages of space are used as background. This is acceptable for small trees, and such diagrams produce excellent results due to their intuitive perceptibility. But for large trees, they are hard to navigate by virtue of their large sizes. Another problem with node-link diagrams is the lack of content information since presenting additional information will quickly overwhelm the display space [7].

C. Novel systems

Novel systems were designed to provide alternative solutions to hierarchy visualisation.

Cone Trees [12] represent large hierarchies in 3D space in an attempt to increase the number of nodes that could be represented on screen. The Hyperbolic Browser [8] combines the focus+context technique with hyperbolic geometry. A hierarchy is drawn uniformly on the hyperbolic plane. Cheops [2] uses a small amount of reusable triangles to represent a large amount of nodes of a hierarchy with triangles lower than the second level overloaded. The H3 browser [10] optimises the layout algorithm of Cone Trees to 3D hyperbolic space by placing children on a hemisphere around the cone mouth instead of on its perimeter. It claimed to be able to visualise a huge graph with over 20,000 nodes. NicheWorks [14] is a recent node-link based system, which

claims to be capable of visualising 100,000 nodes on personal computers and one million nodes on powerful graphics workstations. Other excellent systems include Treemaps [7], and fvis [4]. However, most of these systems emphasise one aspect of a hierarchy, which is structural information. One exception is Treemaps as stated by Johnson "our approach is best suited to hierarchies in which the content of the leaf nodes and the structure of the hierarchy are of primary importance...". Treemaps concentrate on the content information of the leaf nodes. Albeit its ability to depict structural information, it has to be parsed by users themselves with cognitive overhead.

Visual analysis is an important task in the area of Information Visualisation. As indicated by Card, Mackinlay and Shneiderman [3], the perceptual analysis of dynamic information displays is an unsolved problem. "It is as important to understand the nature of information-intensive tasks as it is to understand the details of information visualisation technology. Otherwise, the field will evolve into a set of techniques for making pretty pictures looking for a use".

The new FlexTree approach aims to effectively provide both structural information of the whole hierarchy and dynamic access to content information of any node in the same view, and be able to scale to handle large hierarchies. As such, it is hoped that FlexTree will be the basis of an effective visual analysis tool for hierarchies.

III. FLEXTREE APPROACH

This section gives a description on how to view 2D tree representations, how the FlexTree is constructed and what functionality it can provide.

A. Perspectives of 2D trees

Hierarchies can be drawn entirely in 3D graphics, like the Cone Trees, but in 2D graphics, different views of a tree could generate different presentations. Generally speaking, a real tree may be viewed from three perspectives, bottom-up view, top-down view and lateral view. If an abstract tree is viewed in the same or similar ways, with further transformation, it could output useful visualisations. For example, viewing a tree in a lateral way could lead to a traditional node-link diagram. Normally the tree may also need to be rotated by a certain degree. Viewing a tree from bottom-up and rotating nodes from root to leaves equally could provide a radial view and a balloon view of the tree. Viewing a tree from top-down and letting leaf nodes divide the projection area according to certain rules, thus producing the Treemap. Although the Treemap is not initially obtained from this metaphor, different metaphors could achieve the same arrangement. The FlexTree is a variation of the lateral views from this perspective.

B. Aesthetic rules for design

To effectively display a tree on the screen, certain rules must apply. Charles Wetherell and Alfred Shannon [13] and Edward Reingold and John Tilford [11] defined five aesthetic rules to draw a binary tree. They are as follows:

1. A parent should be drawn above its children.
2. Nodes at the same level should lie along an horizontal line.
3. A left child should be positioned to the left of its parent and a right child to the right.
4. A tree and its mirror image should be drawn to reflect each other.
5. A subtree should look the same, regardless of where it occurs.

As stated in [9], "it makes more sense to use horizontal trees to diagram user-interface trees because typically they contain text strings of varying length." So if the first two rules could be extended from top-to-bottom to left-to-right correspondingly, they would be very useful in the FlexTree design. Since the third and fourth rules are restricted to binary trees, they are not applicable. To keep the fifth rule, one has to allocate enough space between tree nodes. However in Information Visualisation designs, screen pixels are so precious due to the large amount of nodes to be presented. So even though the fifth rule is helpful in understanding a tree, it will require modification. Three rules for the FlexTree design are:

1. A parent should be drawn left of its children.
2. Nodes at the same level should lie along a vertical line.
3. A subtree should look similar, regardless of where it occurs.

C. Rearranging traditional node-link diagrams

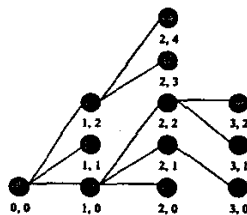


Figure 3. Rearranged node-link diagram with co-ordinates for each node indicating positions

A hierarchy can be organised in such a manner that each node can be coordinated as in Figure 3 so that each node has a unique pair of co-ordinates. The co-ordinates can be mapped onto the X and Y co-ordinates on a Cartesian plane with X-axis standing for level and Y for sequence. In so doing a histogram-like layout may be obtained. In such an arrangement, not only is the number of levels of the hierarchy easily seen but also are the numbers of nodes in each level visually comparable. Thus it is easy to visually identify those levels which have large or small amounts of nodes. In this arrangement none of the relationships within the hierarchy are disposed of. They are represented by lines connecting parent nodes and their children. Although the hierarchy is not as aesthetic as its origin (Figure 2), the relationships are still clearly visible. However using lines is not enough to truly depict the tree structure. For example, the line representation would not permit the users to see how many child nodes

belong to a certain parent node, or to visually compare the numbers of child nodes from two different parent nodes. The first case can be solved by providing a visual cue to highlight all the children of the currently elected node. As to the second case, nodes need to be grouped according to their parents and thus groups separated in a certain manner. The FlexTree solution is to insert a gap (several pixels of blank space) between two nodes from different parents. As the gaps are consistently inserted, the tree structure is much clearer than before. Since different people have different visual ability to recognise structures, the height of the gap is also variable. The bigger the gap's height, the clearer the tree structure. The effect can be seen in Figure 4.

As stated earlier in this paper, a hierarchy always has structural information and content information. Any hierarchy has structural information, but content information is always specific to the domain where the hierarchy is extracted. How FlexTree presents the structural information is described above. In order to express the concept of presenting the content information, it is necessary to provide an application in a specific domain. In this instance the example given is a computer file system.

A computer file system is organised as files and file folders; together they compose the file system hierarchy. In the FlexTree design, files are taken as content information of folders other than leaf nodes. In such a way, only folders contribute to the structural information, files only contribute to the content information of nodes. The structural information is shown explicitly on the screen with the content information accessible through interaction. This is similar to showing only branches of a tree and hiding the leaves. Such a metaphor makes it possible to visualise large hierarchies without letting the huge amount of content information overwhelm the screen.

D. Visual encoding

Encoding information into visual cues is vitally important in visualisation design. A good design is able to speed information understanding and assimilation, whereas a poor design may be of no use in aiding understanding and may even be misleading. It is not always true to say that the more information shown on screen the better the visualisation. One attribute of objects needs at least one visual cue to encode. Many visual cues shown simultaneously could cause cognitive overload, because the mapping from attributes to visual cues has to be borne in mind by users while viewing the visualisation.

Visual cues include length and height, colour, texture, shape, size and symbols etc. Considering the quantity of information to be visualised, it requires careful choice of visual cues. The visual cues must be scalable, which means they should be effectively discernible even when they are only several pixels. FlexTree utilises length, colour, and texture as the main visual cues because they are discernible even when scaled down to one pixel if used properly.

FlexTree uses bars of equal length to stand for nodes. The equal lengths of the bars guarantee that there is no line

crossing between any two neighbour nodes. Each bar has a coloured bar on top of the bar. The colour of the coloured bar is used to encode content modified history of folders when applied to visualising file systems. The colour scheme is visually linearised [5] and selectable by users. Modifying a file contained in a folder would cause the change of the folder's colour.

The length of the coloured bar represents the amount of the content inside the folder, it can be the total size of files or the number files. Adding new files to a folder or deleting existed files from a folder would cause the change of the coloured bar's length.

A three-state-icon is introduced as an indicator of a folder's states, that is open, closed, childless. The icon is allocated at the very right of every node, the node tail. When a folder is childless the icon shows no texture and the same colour as the background of the node. When a folder has children and all the children are visible (open state), the icon shows as a small square with a minus symbol in the middle. When a folder has children and all the children are invisible (closed state), the icon shows as a small square with a plus symbol in the middle. This encoding is similar to Windows Explorer to some extent; this is designed in order to take advantage of the familiarity of such encoding. Another benefit of this encoding is that the state of a node is still visible even when the height of nodes reduces to one pixel.

FlexTree does not encode much meaning with the links; they only represent parent-child relationships. The colour of lines changes when a node is browsed or selected. Browsing or selecting a node will highlight the lines linked to it to emphasise all the related nodes.

E. Interaction

Interactive systems provide a good means of dealing with large quantities of information. It is ubiquitous in current visualisations. One of the main benefits of applying interaction is that it can temporarily hide the rest of the information while showing one part of it. It is also easy to achieve different views of the same data set through interaction. In the design of FlexTree, these two benefits are fully utilised.

Interacting with the FlexTree system is easy. Some representative functions are illustrated in more detail.

1) Navigation

As Beard states [1] on navigating large 2D spaces, "If the two-dimensional information space fits completely onto a display screen, there is no navigation problem ... Users are never lost because they can see the complete information space." Since the FlexTree system is a 2D visualisation, this goal has been kept in mind in the design. Scrolling is applied to the navigation of the FlexTree. Therefore the scalability becomes an issue. It is difficult to provide an overview of too many nodes. It is recommended that an overview should at least be able to see one quarter of the scene.

2) Zooming

Zooming is simple for users to understand, but users lose the overview after they have zoomed in and the mechanism for

zooming out is not always apparent [3]. Zooming is an important part of the interaction with the FlexTree. A strategy was carefully designed to overcome the above problem. In this strategy, vertical zoom and horizontal zoom are separated. Horizontal zoom only works on levels of nodes. It uses the same mechanism as operating a usual database table or electronic spreadsheet. When users want to see more details of a level of nodes or some details of a single node in that level, they can drag the border of the column to the right to zoom in and to the left to zoom out. Vertical zoom is further divided into node-zoom and structure-zoom. Node-zoom has effect only on nodes; it can be performed on all nodes, a level of nodes and individual nodes. All the global functions in the FlexTree are executed through clicking buttons including the vertical node-zoom. A pair of vertical zoom buttons is specially designed and they are metaphorised by cylinder lenses employing the common metaphor "+" to zoom in and "-" to zoom out. Vertical zooming with a level of nodes and an individual node uses the same mechanism; they utilise the mouse wheel. Scrolling the wheel forward to zoom in and backward to zoom out. The only difference between them is the position of the mouse cursor. For individual node operation, the cursor needs to be positioned on the node, whereas for level of nodes the cursor needs to be on the background of the column. Structure-zoom is even easier. Left-click on the background of a column will further separate nodes from different parents. Right-click is to minimise the separation.

So far two pairs of counter concepts are employed in the zooming design, that is right and left, forward and backward. It results in a clear mechanism for the zooming operation. Once users are familiar with one operation the remainder appear natural.

Another aspect worth mentioning is that the node zooming is semantic zooming in concept because the information content changes and more details are shown when approaching an area of interest [6]. It is actually achieved by gradually showing the whole of the underlying picture starting from a small part or vice versa. The authors believe that this mechanism has generic potential to other visualisation designs.

3) Searching

Considering the size of the information space, to find points of interest only by browsing normally would not be enough. For fast access of points of interest a search function is also provided for the FlexTree. Performing a search will cause the matched points to be highlighted. Currently, search is only supported by name. Node matching and content matching are differentiated by highlighting using a traffic lights mechanism. If node names match the search term, the matched nodes are highlighted by red. If nodes content match the term, the nodes are highlighted by green, and if both highlighted by yellow.

4) Sorting

Ordered information is always easier to understand. Sorting is an important feature of the FlexTree system. This function is provided in order to facilitate navigation of the hierarchy and compare related nodes according to certain attributes. Sorting is divided into global sort and level sort to enhance performance. Global sort has effects on all nodes in the

hierarchy, whereas level sort only affects certain levels. Sort by any attributes of nodes is supported. However sorting a tree is not like sorting a table of records in a database. Because trees are already structured, it is not possible to keep the tree structure while being able to sort all nodes in a level. The reasonable solution is to only sort nodes from the same parent, but also to be considered the rearrangement of the lower level nodes to avoid line crossing. The sorting function always give visual feedback to users, either in colour or in bar length (the two main visual cues used to encode information). The encoding from attributes to visual cues is selectable. This makes the sorting versatile. Through sorting more insight can be gained into the hierarchy.

IV. IMPLEMENTATION AND TESTING

The FlexTree system was developed as a stand-alone application using the Java Development Kit version 1.4 Beta (JDK 1.4 β), to utilise the mouse wheel. The performance of the system has been tested on different environments. On a personal computer (PC) with 128 MB main memory and 4MB video memory, Pentium 500MHz CPU, and Windows 98 as the operating system, a computer file system hierarchy with 6,351 directories and 130,400 files was successfully visualised. The overall performance is a little slow, but acceptable. With a better PC having 256MB memory and Pentium 800MHz CPU, both the performance and the scalability is much improved. Two screenshots were taken and are shown in Figure 4 and 5. In Figure 4, two local folders were zoomed in to show more details while keeping the other folders in their minimum states as context. The vertical gaps were also magnified to show grouping in level 8. Figure 5 shows the overview of the whole hierarchy of a PC's D: drive with JDK1.3 being browsed.

The testing data is simply retrieved from a working PC by using Disk Operating System (DOS) "dir" command. There is one issue worth mentioning. The DOS *directory* and *folder* are two different concepts. For example, in Figure 6 the total number of nodes (folders) is 6,351, but the number of directories is 19,047 shown by the results of "dir" command. The mismatch is caused by the fact that the "dir" command counts twice the current and parent folder for each file folder.

V. FUTURE DIRECTIONS

The usability of a system is always critical to its success or failure. It is intended to carry out comprehensive usability testing on the FlexTree as part of the future work. Testing in other domains other than computer file systems will also be considered for instance the potential of utilising FlexTree as a technique for visualising user profiles on accessing web pages. Graph visualisation is an open research area yet of many existing difficulties, discussion of which are beyond the scope of this paper. A tree is the simplest form of graph. If a tree can be extracted from a graph, it will aid the understanding of the graph. It is proposed to combine the FlexTree into a more complex graph visualisation system.

VI. CONCLUSION

The main underlying principles in designing this prototype are:

- Miniaturising data points into their minimum size whereas still placing discernible and easily comparable attributes on them to achieve maximum screen pixel utilisation.
- Simplifying manipulation to maximise ease of use.
- Organising every picture to gain high visual clarity and intuition.
- Always keeping context information to prevent users from disorientation.
- Reducing high computational constraints to obtain high performance.

It is hoped that these principles are also beneficial in general Information Visualisation design, not only the FlexTree system.

VII. ACKNOWLEDGEMENTS

This project is fully sponsored by Nortel Networks (Northern Ireland) Jigsaw Research Programme. Hereby we thank Chen Chen, a MSc student in Faculty of Informatics, for his industrious work on part of the implementation of the system. We also thank our colleagues within Jigsaw Strand 4 for their comments and feedback.

VIII. REFERENCES

- [1] Beard, D. V. and II, J. Q. W., Navigational techniques to improve the display of large two-dimensional spaces, *Behaviour and Information Technology*, 1990, 9 (6): 451-66.
- [2] Beaudoin, L., Parent, M.-A. and Vroomen, L. C., Cheops: A Compact Explorer for Complex Hierarchies, in *Proc. IEEE Visualization '96*, pp. 87-92, San Francisco, USA, October 27 - November 1, 1996. Computer Society Press.
- [3] Card, S. K., Mackinlay, J. D. and Shneiderman, B., Eds, *Readings in Information Visualization: Using Vision to Think*. The Morgan Kaufmann Series in Interactive Technologies, Morgan Kaufmann, San Francisco, 1999. ISBN: 1-55860-533-9.
- [4] Carriere, J. and Kazman, R., Interacting with Huge Hierarchies: Beyond Cone Trees, in *Proc. IEEE InfoVis '95*, pp. 74-81, Atlanta, USA, October 30-31, 1995. Computer Society Press.
- [5] Chalmers, M., Undergraduate Course Lecture Notes on Interactive Systems, from a tutorial on information visualisation at *VLDB'99* conference, Department of Computing Science, Glasgow, UK, 1999.
- [6] Herman, I., Melancon, G. and Marshall, M. S., Graph Visualisation and Navigation in Information Visualisation: a Survey, *IEEE Transactions on Visualization and Computer Graphics*, 2000, 6 (1): 24-43.
- [7] Johnson, B. and Shneiderman, B., Treemaps: A Space-Filling approach to the visualization of hierarchical information structures, in *Proc. IEEE Visualization '91*, pp. 284-291, San Diego, California, USA, October 22-25, 1991. IEEE Computer Society Press.
- [8] Lamping, J., Rao, R. and Piroli, P., A focus + context technique based on hyperbolic geometry for visualizing large hierarchies, in *Proc. ACM CHI '95*, pp. 401-8, Denver, Colorado, USA, May 7-11, 1995. ACM Press.
- [9] Moen, S., "Drawing Dynamic Trees" *IEEE Software*, 1990, 7 (4): 21-28.

[10] Munzner, T. H. 3: Laying out large directed graphs in 3D hyperbolic space. In *Proc. the IEEE Symposium on Information Visualization '97*, pp. 2-10, Phoenix, USA, 1997.

[11] Reingold, E. M. and Tilford, J. S., Tidier Drawings of Trees, *IEEE Trans. Software Eng.*, pp. 223-228, September, 1981.

[12] Robertson, G. G., Mackinlay, J. D. and Card, S. K., Cone Trees: Animated 3D Visualizations of Hierarchical

Information, in *Proc. CHI '91: Human Factors in Computing Systems*, pp. 189-194, New Orleans, Louisiana, USA, April 27 - May 2, 1991. ACM Press.

[13] Wetherell, C. and Shannon, A., Tidy drawings of trees, *IEEE Trans. Software Eng.*, pp. 514-520, September, 1979.

[14] Wills, G. J., NicheWorks: interactive visualization of very large graphs, in *Proc. Graph Drawing '97*, pp. 403-414, Rome, Italy, September 18-20, 1997. Springer-Verlag, Berlin.

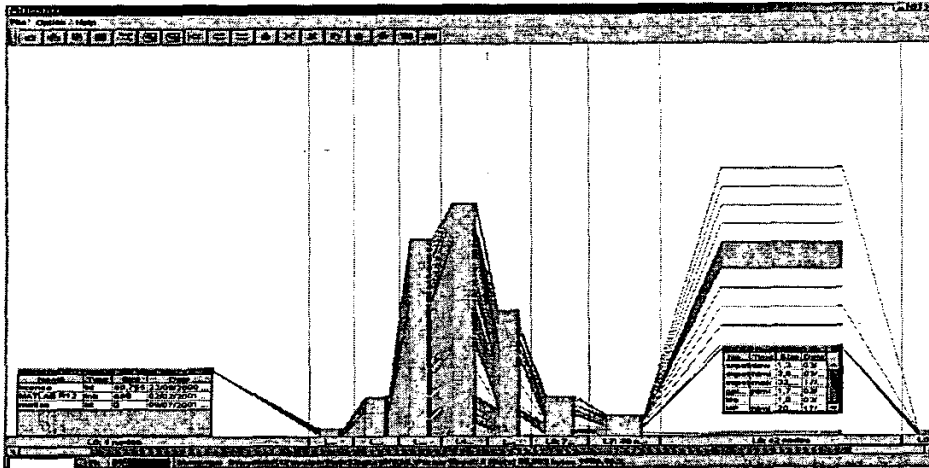


Figure 4. The screen shot of the FlexTree visualising the Matlab package. The root directory and a directory called Expat were selected as foci. Gaps were added for level 8.

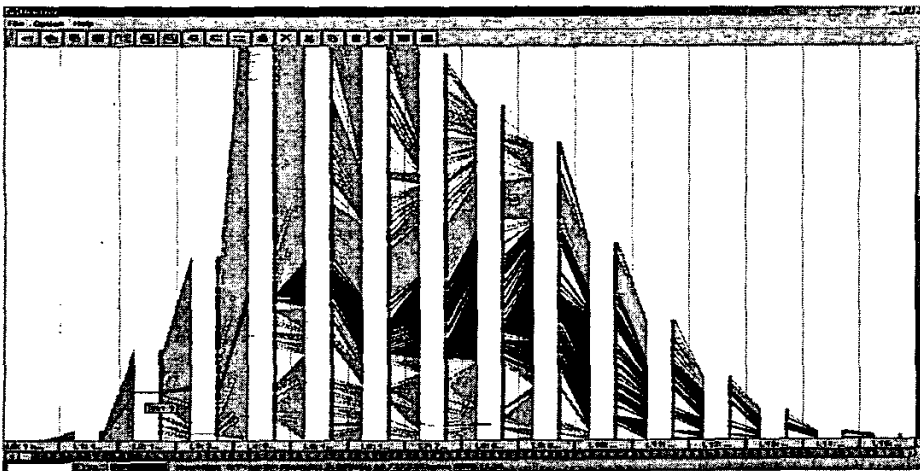


Figure 5. The screen shot of the FlexTree visualising a file system with 6,351 directories and 130,400 files, the total size of all the files is 6,936,747,234 bytes.